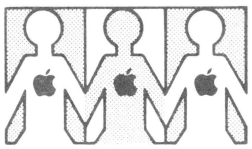
contact issue no.1

the user group newsletter apple computer inc. may 1978



LOCAL USER GROUPS

Apple is springing up all over

Local APPLE II User Groups have begun to spring up all over the world. In an effort to support them, we will publish information on the ones we know about. If you are part of a club that isn't mentioned here, drop us a line and we'll tell the world about you. If there is no group in your area and you want to start one, talk to your APPLE dealer. Chances are he knows most of the local APPLE II owners, and can help bring them together for your kickoff meeting. He might even want to host the meetings on a continuing basis. Ask him.

Here are the groups we know about:

APPLE USER GROUP
CARR ELECTRONICS CORP.
5811 Geary Blvd.
San Francisco, CA 94121
Bruce Tognazzini
(415) 668-4243

APPLE PUGET SOUND PROGRAM LIBRARY EXCHANGE
6708 39th Ave. Southwest
Seattle, Wash., 98136
Val Golding
(206) 937-6588 (home)
(206) 623-7966 (work)

APPLE CORE
AVIDD ELECTRONICS
2210 Bellfower Road
Long Beach, CA 90815
(213) 598-0444

APPLE USER GROUP
DATAMART INC.
3001 N. Fulton Drive
Atlanta, GA 30305
Preston Love
(404) 266-0336

In addition to the above groups which are oriented around the APPLE II, there is an APPLE I User Group. For information write to:
Joe Torzewski
51625 Chestnut Road
Granger, IN 46530

LOOKING AHEAD

Curing the Apple II itch

In case you are itchy to add to your APPLE II system (or even just to learn more about it), here's what's coming up.

BASIC PROGRAMMING MANUAL-This is a beginner's guide to the Apple BASIC complete with lots of examples and illustrations. It will be mailed to all users who have warranty cards on file with us, starting late this month.

COMMUNICATIONS CARD-A card that lets your APPLE II talk over the phone with other computers will become available in April. For $180, this intelligent interface will allow the computer to control any 110 or 300 baud serial device through an industry standard RS-232C interface port. Look for a data sheet in the next newsletter.

NEW APPLESOFT-It's coming, and it's going to be great! The new APPLESOFT fixes bugs in the existing version, and adds features such as data save/load, high-resolution plotting capability, ON ERR GOTO capability, and much more. It wil be available on cassette in May at $20. The ROM version, on a plug-in card (for slot #0), will be out in June at $99. The card will have a switch to allow you to choose between Apple BASIC and APPLESOFT.

FLOPPY DISK-The new mini-floppy should be on dealers' shelves by early July, at a price of less than $700 for the controller card and one drive. (Each card will handle up to two drives.) The software, which works with either version of BASIC, will be able to load and store named files and provide disk directory lists. Look for more details two issues from now.

SERIAL INTERFACE CARD-This is a high speed (to 9600 baud), programmable serial interface; designed to connect fast, half-duplex devices (printers, plotters, etc.) to the APPLE II. More details will be available two issues from now, and the device itself will be out in July. No price has been set.

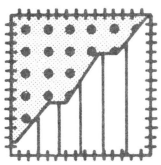
EDITORIAL by Phil Roybal, Marketing Mgr.

A funny thing happened on the way here . . . .

This is the first of an irregularly published series of newsletters aimed at filling you in on what's happening at Apple; and at soliciting the feedback, product ideas, and people we need to service you, our customer base. In the past, we occasionally let our enthusiasm carry us away. It is often easy to forget how difficult it is to actually bring quality products into production. (And that's especially true when you're doubling in size every few months and still can't hire people fast enough to do everything that needs doing.) But on the way here, we've learned a bit. And we realize success hinges upon meeting commitments. So from now on, you'll see more conservative schedules from us; and we'll usually beat them.

This newsletter marks the start of an official APPLE II User Group: one that we trust will grow stong and healthy this year. But there's a lot of work yet to do. A major portion of it has to do with classifying and documenting all the software we have received over the past months. It really hasn't been lost. All 200 tapes are sitting in a box, waiting for us to look at them. While our goal was to publish a software list, description of the Software Bank, and Contributor's Form with this newsletter, we started too late. Instead, it'll be in the next newsletter. For sure.

Apple Computer has grown from two men in a garage to 40 people in a modern 20,000 square foot building, in less than two years. You made that possible, and we thank you. In the coming year we will continue to earn your support; and to actively solicit your inputs so that our efforts stay on target. It is important to us. We learned that on the way here.



PATCHES AND PROGRESS

Too many gosubs

From time to time, we turn up a program bug. Although the fixes are incorporated into future versions of the program, often they are simple enough that users can "patch" their present programs to get improved performance. Here are patches for two popular programs: CHECKBOOK and APPLESOFT.

CHECKBOOK CHANGES-These changes will make the CHECKBOOK program more efficient, as well as eliminating the "TOO MANY GOSUBS" error message that occasionally crops up. Changed sections are underlined.

APPLESOFT CHANGES-The following patches to APPLESOFT will fix problems associated with the FRE, END, and DIM statements; and will allow the language to handle long program lines. The changes are made with a series of POKEs. They can be done from the command mode, or incorporated right into your program.

CHANGES

ARRAY INDEXING PROBLEM FIX:

POKE 6331,32: POKE 6332,150: POKE 6333,41: POKE 6334,234
POKE 10646,133:POKE 10647,177: POKE 10648,162: POKE 10649,5: POKE 10650,165: POKE 10651,132: POKE 10652,96

LONG LINE FIX:

POKE 3050,234
POKE 3054,136
POKE 3055,145
POKE 3056,158
POKE 3057,208
POKE 3052,251

'END' STATEMENT FIX:

POKE 2048,210

FRE () FUNCTION FIX:

POKE 6143,5

APPLESOFT MANUAL CORRECTION-a small typesetting disaster left us with an incomprehensible example on page 22 of the APPLESOFT Manual. The example should read as shown on page 4.

CHECKBOOK Program Changes

LINE # FROM

```
LIST 0,32767
 1111 PRINT ^
 1221 C=S1: VTAB L: CALL -958: IF
      C<Z THEN 1100: IF C>9999 THEN
      1225: D(0)=C: E=4: GOSUB 7: GOTO
      1230
 1360 PRINT "FROM CHECK #";D(5);" TO C
      HECK #";D(6): GOSUB 9: CALL
      Z: BALI=D(0): BALF=D(2): RBI=D(
      3): RBF=D(4): RETURN
 1629 IF RBF<-99 THEN RBI=RBI-0: IF
      RBF>Z THEN RBI=RBI-0: RBF=(RBF-
      100) MOD 100
 1810 GOSUB 6: T=12: T1=32: FOR L=3
       TO 17: IF P>=CM THEN 1840
 1815 GOSUB 30
 1860 IF L$="" THEN 1810: IF L$="R"
       THEN 1100: IF L$="M" THEN
      2440: IF L$="L" THEN 1870: VTAB
      L: TAB 0: GOSUB 22: GOTO 1842
 2250 T=12: T1=32: VTAB L: TAB 0: GOSUB
      2: GOSUB 11: SL=B: B=T: GOSUB
      5: PRINT " ";C$;: L=L+0: P=P-
      S-S: BALI=BALI+R(10): BALF=BALF+
      R(11): GOSUB 1427: RETURN
 2296 GOTO 1100
 2620 D(0)=BALI: D(2)=BALF: D(3)=RBI:
      D(4)=RBF: P=LM: GOSUB 2: INPUT
      "START RECORDING, THEN HIT RETUR
      N",L$
```

TO

```
LIST 0,32767
 1111 PRINT :H=0
 1221 C=S1: VTAB L: CALL -958: IF
      C<Z THEN RETURN : IF C>9999
       THEN 1225: D(0)=C: E=4: GOSUB
      7: GOTO 1230
 1360 PRINT "FROM CHECK #";D(5);" TO C
      HECK #";D(6): GOSUB 9: CALL
      Z: RETURN
 1629 IF RBF<-99 THEN RBI=RBI-0: IF
      RBF>Z THEN RBI=RBI+0: RBF=(RBF-
      100) MOD 100
 1810 GOSUB 6: GOSUB 30: T=12: T1=32
      : FOR L=3 TO 17: IF P>=CM THEN
      1840
 1815 REM  DELETE THIS LINE ENTIRELY
 1860 IF L$="" THEN 1810: IF L$="R"
       THEN RETURN : IF L$="M" THEN
      2440: IF L$="L" THEN 1870: VTAB
      L: TAB 0: GOSUB 22: GOTO 1842
 2250 T=12: T1=32: VTAB L: TAB 0: GOSUB
      2: GOSUB 11: SL=B: B=T: GOSUB
      5: PRINT " ";C$;: L=L+0: P=P-
      S-S: BALI=BALI+R(10): BALF=BALF+
      R(11): GOSUB 1427
 2251 IF L<23 THEN RETURN
 2252 INPUT "TYPE 'R' TO RETURN TO THE
       MENU, 'RTN' TO CONTINUE SEARCHI
      NG",C$: IF C$="R" THEN RETURN
      : GOTO 2235
 2296 RETURN
 2620 P=LM: GOSUB 2: INPUT "START RECO
      RDING, THEN HIT RETURN",L$
```

```
 100 DIM A$(15)
 110 FOR I=1 TO 15:READ A$(I):NEXT I
 120 F=0:I=1
 130 IFA$(I)<=A$(I+1) THEN 180
 140 T$=A$(I+1)
 150 A$(I+1)=A$(I)
 160 A$(I)=T$
 170 F=1
 180 I=I+1:IF I<15 GOTO 130
 190 IF F=1 THEN 120.
 200 FOR I=1 TO 15:PRINT A$(I):NEXT I
 220 DATA APPLE, DOG, CAT,RANDOM,COMPUTER,BASIC
 230 DATA MONDAY,"***ANSWER***","FOO: "
 240 DATA COMPUTER,FOO,ELP,MILWAUKEE,SEATTLE,ALBUQUERQUE
]
```

Problems and solutions

Part of a personal computer's charm lies in the fact that it is a creative tool. Each person uses it differently, to accomplish different goals. Unfortunately, this makes it difficult to write a manual that adequately covers everything a person might want to do with his system. The HOW TO section is therefore devoted to answering questions the manuals missed.

LOADING MACHINE LANGUAGE AS PART OF A BASIC PROGRAM

Often we want to include machine language data inside a BASIC program. A great many Apple tapes are made up this way to simplify the loading process. Here's a recipe for doing it yourself with programs written in Apple BASIC.

Apple BASIC loads programs into memory with the highest program line at the highest RAM location (HIMEM). Preceeding lines are located lower and lower in RAM. The beginning of the program is at PP, an address which is held in memory locations CA and CB (hexadecimal), or 202 and 203, decimal. When you type SAVE, the computer transfers to tape everything between PP and HIMEM. Thus, to tuck machine language into your program so that it can later be loaded like BASIC, it is merely necessary to move the PP pointer down below the beginning of the extra code, put in two POKES to reset the pointer before running the program, and type SAVE. Later, you will be able to LOAD the whole thing just as if it were all BASIC. Just follow these steps:

1. Get the BASIC program into memory, just the way you want it. If you make any changes, you must re-do steps 2 and 6.
2. In the command mode, type:
   PRINT PEEK (202), PEEK (203) and write down the results. Let's call them m and n, respectively.
3. Load your machine language code into memory using the monitor load capability (xxxx.yyyyR). This will put the machine language program into memory below the beginning of the BASIC program, starting at hexadecimal address xxxx.
4. Take the starting address of the machine language program and divide it into two parts: xx/xx. Convert each pair of digits from hex to decimal values: a & b; corresponding to the left and right xx pairs, respectively. Write them down.
5. Now enter the BASIC command mode and type:
   POKE 202, b-1 (value **b** from step 4, above)
   POKE 203, a (value **a** from step 4, above
   POKE 204, 00
   POKE 205, 8
6. You have now moved the pointers down below your machine language program, and must insert code to move them back again when the program

is run. To do that, type:
0 POKE 202,m: POKE 203,n: GOTO q
where m and n are the values from Step 2, and q is the first line number in your BASIC program. That line number can be 0—it will not be erased by the above entry.

7. Now you're done! Don't try to list your program before running it, because all you'll see is a meaningless set of numbers and symbols. Just type SAVE (before running the program), and it will all go onto tape. Late a LOAD command will bring it all back in.

CAUTION!

Once you have RUN such a program, you cannot SAVE it, for the pointers will have been moved. You can only save or copy a program like this before it has been RUN.

PRINTING LOWER-CASE LETTERS WITH APPLE II

APPLE II cannot presently display lower-case characters on the screen, but it has no trouble printing them on most printers. To create a string of lower-case characters, simply generate the string in upper case at a known memory location; and then go through and add 32 to the ASCII value of each upper-case letter. That will produce the lower-case ASCII equivalent. If you then shove the modified code back into the string variable in place of the old value and print the string, you will print lower case characters.

As you convert the string, you must test each character to see that it is not a numeral, punctuation, etc. Obviously, only alphabetic characters can be converted to lower-case. Here's a sample program that does the job.

This program works because, by defining A$ first, we know its absolute address in memory. Its first letter is stored in address 2053. (If this program were to be converted to APPLESOFT, we would have to manipulate our characters entirely through the string routines, since we cannot accurately locate the string in memory.)

Lower Case Character Generating Program

```
LIST
    10 DIM A$(80): REM  BUFFER...MUST BE FIRST VARIABLE DEFINED IN PROGRAM
    20 INPUT "ENTER DATA";A$
    30 FOR I=1 TO LEN(A$)
    40 C= PEEK (2052+I): REM  SET C=ASCII VALUE OF "I TH" CHARACTER OF A$
    50 IF C<193 THEN 70: REM    TEST FOR A NON-ALPHABETIC CHAR. IF FOUND, DO NOTHING.
    60 POKE 2052+I,C+32: REM  CONVERT LETTER TO LOWER CASE AND INSERT BACK INTO A$.
    70 NEXT I
    80 CALL -936: REM  CLEAR SCREEN
    90 PR#1: REM  TURN ON PRINTER
   100 PRINT "": REM  CTRL/I 80 N SETS LINE LENGTH TO 80 CHAR.
   110 PRINT A$
   120 PRINT "": REM  CTRL/I 40 N RESET
       S LINE LENGTH TO 40.
   130 PR#0: REM  TURNS OFF PRINTER.
   140 END

>
```

```
RUN
ENTER DATATHIS IS AN EXAMPLE OF HOW TO PRINT LOWER-CASE LETTERS.

this is an example of how to print lower-case letters.
```

OUT OF THE MIST ....

Making sense out of wonderful features that no one understands

Occasionally we document some wonderful feature in such a way that nobody understands it. As we identify those areas (recognizable by the stack of associated phone messages), this column will attempt to clarify them.

24K SYSTEM PROGRAM LOADING

Normally, entering BASIC with the B^{C} command resets the pointers to high and low memory (HIMEM & LOMEM), so that the monitor knows where to begin loading programs. However, 24K systems need a little human help to set the HIMEM pointer. Therefore, before loading or keying in programs, enter BASIC in the command mode and type:

HIMEM:24576

When this step is omitted, the system believes it has a 32K memory to work in. Since BASIC programs are loaded starting from the top of memory, they tend to fall into the bit bucket and disappear forever . . . .

PROGRAM LISTING WITH THE TTY PRINTER ROUTINE

The hardware/software TTY interface described on page 114 of the APPLE II REFERENCE MANUAL provides an inexpensive method of printing on a TTY (output only). The example on page 116 shows how commands can be incorporated into a BASIC program to produce hard copy output.

Unfortunately, we left out an example of how to LIST a program on the TTY. Here's how. Just enter the command mode of BASIC and type:

```
CALL 880
LIST
PR #0
```

That sequence will enable the TTY, list the program, and then return output to the TV screen.

USING THE HI-RES ROUTINES

A section of the new REFERENCE MANUAL describes the HI-RES plotting routines and mentions that they are available in ROM and on tape. Although the ROM's aren't available yet, many people don't know that they have the routines on tape already. These programs are the machine language load at the start of the HI-RES DEMO tape. If you load the machine language part and then skip loading the BASIC demo program, you will have the routines in memory to work with.

While use of most of the programs is straightforward, the SHAPE routine gives some people problems because they don't see how to build and use a shape table. Here's an explanation.

The SHAPE routine reproduces a figure from a set of instructions (the shape table) stored somewhere in memory. But it has to know where to find that table. It has been written to assume that the shape table begins at an address which is pointed to by memory locations 804 and 805. What you insert in those locations depends upon where you built your shape table.

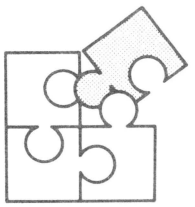
Let's use the table example given in the REFERENCE MANUAL (page 53) to illustrate how to build and use the table. Here are the steps to follow:

1. Load the HI-RES routines into memory (C00.FFFR).
2. Build the shape table in memory, using the monitor. Let's arbitrarily start at address 900 hex, and fill in the data sequentially, just as shown in the manual: *900:12 3F 20 64 2D 15 36 1E 07 00
   This will put the table above BASIC's variable space and below the bottom of a BASIC program.
3. Insert the shape table starting address into memory locations 804 and 805, as shown in line 10 of the sample program that follows. (For more details, see the explanation in the HOW TO section under LOADING MACHINE LANGUAGE PROGRAMS . . . .). Then insert information on color, scale, rotation, etc., into the other memory locations specified in the REFERENCE MANUAL.

A BASIC Program Using a Shape Table

```
LIST
    5 CALL 3072: REM   INITIALIZE HI-RES ROUTINES
   10 POKE 804,0: POKE 805,9: REM  SHAPE TABLE POINTERS
   20 POKE 28,170: REM   COLOR CHOICE
   30 POKE 806,1: REM    SCALE FACTOR
   40 POKE 807,0: REM  ROTATION FACTOR
   50 POKE 800,100: POKE 801,0: REM   SET X LOCATION
   60 POKE 802,100: REM   SET Y LOCATION
   70 CALL 3761: REM  POSITION CENTER OF PLOT
   80 CALL 3805: REM   CALL SHAPE ROUTINE
   90 END

>
```



BITS AND PIECES

New modulator means no more wavy lines

Some APPLE II users have noticed wavy lines or color patches on their TV screens caused by radiation from airplanes, motors, or the computer itself. A UHF modulator which operates at Ch. 33 instead of Ch. 3 seems to solve these problems neatly. The new device, called the Sup-R-Mod II, will be available in April at $29.95, from

M&R ENTERPRISES
P.O. Box 1011
Sunnyvale, CA 94088

The new modulator comes completely assembled, ready to plug into the APPLE II.

Calling all interface card designers

Now that several other companies have begun to market interfaces for the APPLE II, a word on our design philosophy is in order.

We are very interested in any devices that plug into APPLE II, and are generally happy to provide information to prospective manufacturers. (Just write and ask for our Prototyping Board Write-up). Our own philosophy is to build devices that are intelligent (have all control routines in on-board ROM) and are slot independent. However, we must sometimes compromise because of design considerations in the system.

Therefore, we have made the following restrictions on slot assignments:

SLOT #	**ASSIGNMENT**
0	APPLESOFT BASIC ROM Card
6	Second Disk Controller Card
7	First Disk Controller Card

In addition, slots 4 and 5 are tentatively reserved for products that may have to be slot-dependent. Therefore, other manufacturers should design peripherals which are either completely independent of slot number, or will work in slots 1-3. In this way they will avoid possible conflicts with existing or proposed Apple Computer products.

apple computer inc.®
10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010